Feature Extraction in Machine Learning: Turning Raw Inputs into Useful Representations

Ahmed BADI
ahmedbadi905@gmail.com
linkedin.com/in/badi-ahmed

January 2026

Abstract

Feature extraction is a key step in modern machine learning pipelines. Instead of directly using raw inputs or manually engineered variables, we transform the data into a new feature space that is more compact and more informative for the learning task. This article introduces the main ideas behind feature extraction, explains how it differs from feature selection, and presents several important techniques, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and autoencoder-based representation learning. For each method, we provide an intuitive explanation, the core mathematical formulas, and practical remarks on when and why to use it. We also discuss the benefits and limitations of feature extraction, and how it fits into real-world machine learning workflows.

Keywords: Feature Extraction, Dimensionality Reduction, PCA, LDA, Autoencoders, Representation Learning, Machine Learning.

1 Introduction

In many real-world problems, raw data is high-dimensional, noisy, and difficult to work with directly. Examples include images (thousands of pixels), audio signals, sensor data, or text represented as word counts. Training models directly on such raw features can lead to high computational cost, overfitting, and poor generalization. [1], [2]

Two main strategies address this issue: **feature selection** and **feature extraction**. Feature selection chooses a subset of the original features, keeping their meaning. Feature extraction, in contrast, *creates new features* by transforming or combining the original ones. These new features live in a different space, often lower-dimensional, and are designed to capture the most important structure in the data. [1], [3]

Feature extraction is especially useful when:

- The original inputs are extremely high-dimensional (e.g., images, text).
- Many features are highly correlated or redundant.
- We want to capture underlying latent factors, such as topics in documents or patterns in images.

In this article, we focus on three important families of feature extraction methods:

- **Linear unsupervised methods**, such as PCA.
- **Linear supervised methods**, such as LDA.
- **Nonlinear representation learning**, especially autoencoders.

2 Feature Selection vs Feature Extraction

It is helpful to clearly distinguish between feature selection and feature extraction. [1], [3]

- **Feature selection**: keep a subset of the original features (columns), discard the rest. The meaning of each selected feature is unchanged.
- **Feature extraction**: compute new features as functions of the original ones (often linear or nonlinear combinations). The new features (components, embeddings) do not correspond to single original variables anymore.

Formally, suppose the original input is $\mathbf{x} \in \mathbb{R}^p$.

- **Selection**: choose an index set $S \subseteq \{1, \ldots, p\}$ and use $\mathbf{x}_S \in \mathbb{R}^{|S|}$ as the input to the model.
- **Extraction**: learn a mapping

$$\phi : \mathbb{R}^p \to \mathbb{R}^m, \quad \mathbf{z} = \phi(\mathbf{x}),$$

where $m \leq p$, and use the transformed vector $\mathbf{z}$ as the new representation.

Feature extraction can reduce dimensionality and decorrelate features, but it usually sacrifices direct interpretability, because each extracted feature mixes information from many original variables. [4]

Table 1 summarizes the main differences.

	Feature Selection	**Feature Extraction**
What it does	Chooses subset of original features	Creates new features from original ones
Interpretability	High (same meaning as original)	Lower (combinations, latent factors)
Dimensionality	Reduced by dropping features	Reduced by mapping to new space
Examples	Filter/wrapper/embedded methods	PCA, LDA, autoencoders

Table 1: Feature selection vs feature extraction.

3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a classic linear feature extraction method. It finds directions in the data that capture the largest variance and uses them to define new features called *principal components*. [5], [6]

3.1 Goal and Intuition

Given centered data matrix $X \in \mathbb{R}^{n \times p}$ (each row is a sample, each column a feature), PCA seeks directions $\mathbf{w}_1, \ldots, \mathbf{w}_m$ (with $m \leq p$) such that:

- Each component $z_{i1} = \mathbf{w}_1^\top \mathbf{x}_i$ has maximum variance.
- Subsequent components are orthogonal and capture the remaining variance.

Intuitively, PCA rotates the feature space so that most of the variation lies along the first few axes. By keeping only the first m components, we obtain a lower-dimensional representation that preserves most of the information (variance). [6]

3.2 Mathematical Formulation

Let $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ be the sample mean, and assume data is centered: $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$. The sample covariance matrix is

$$C = \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^\top \in \mathbb{R}^{p\times p}.$$

PCA solves the eigenvalue problem:

$$C\mathbf{w}_k = \lambda_k\mathbf{w}_k,$$

where eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ represent the variance along each principal component, and $\mathbf{w}_k$ are the corresponding eigenvectors.

The first m principal components of a sample $\mathbf{x}_i$ are given by

$$\mathbf{z}_i = W^\top\tilde{\mathbf{x}}_i \in \mathbb{R}^m,$$

where $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m] \in \mathbb{R}^{p\times m}$.

3.3 Dimensionality Reduction

The proportion of variance explained by the first m components is

$$\text{ExplainedVariance}(m) = \frac{\sum_{k=1}^{m}\lambda_k}{\sum_{k=1}^{p}\lambda_k}.$$

In practice, we choose m such that this ratio exceeds a threshold (for example, 90%). Then we project data onto the first m principal components. [5], [7]

3.4 Advantages and Limitations of PCA

Advantages:

- Simple, fast, and widely implemented.
- Decorrelates features and reduces dimensionality.
- Often works well as a first step for visualization and modeling.

Limitations:

- Linear method: may miss nonlinear structure.
- Components can be hard to interpret.
- Sensitive to scaling; features should be standardized.

4 Linear Discriminant Analysis (LDA) as Feature Extraction

Linear Discriminant Analysis (LDA) is often used both as a classifier and as a supervised feature extraction method. Unlike PCA, which is unsupervised and focuses on variance, LDA uses class labels to find directions that best separate classes. [8]

4.1 Between-class and Within-class Scatter

Assume we have K classes. Let $\boldsymbol{\mu}_k$ be the mean of class k, and $\boldsymbol{\mu}$ the global mean. LDA defines:

$$S_B = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^\top$$

as the *between-class scatter*, and

$$S_W = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top$$

as the *within-class scatter*, where N_k is the number of samples in class k, and $\mathcal{C}_k$ is the set of indices for class k.

4.2 Optimization Problem

LDA chooses projection directions that maximize the ratio of between-class to within-class scatter:

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}.$$

The optimal directions are obtained by solving the generalized eigenvalue problem

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}.$$

The top discriminant vectors form a projection matrix W_{LDA}, and the extracted features are

$$\mathbf{z}_i = W_{\text{LDA}}^\top \mathbf{x}_i.$$

LDA can reduce the data to at most $K - 1$ dimensions (number of classes minus one). [8]

4.3 Advantages and Limitations

Advantages:

- Uses class labels to maximize class separability.
- Produces low-dimensional representation useful for classification.

Limitations:

- Assumes linear separability to some extent.
- Requires estimates of class means and scatter; sensitive when data is scarce.

5 Autoencoders and Deep Feature Extraction

Autoencoders are neural networks trained to reconstruct their input, typically through a bottleneck layer of smaller dimension. The activations in this bottleneck form a learned, nonlinear feature representation of the data. [9], [10], [11]

5.1 Architecture

A basic autoencoder has two parts:

- **Encoder:** maps input $\mathbf{x} \in \mathbb{R}^p$ to a latent vector $\mathbf{z} \in \mathbb{R}^m$:

$$\mathbf{z} = f_\theta(\mathbf{x}).$$

- **Decoder:** reconstructs $\hat{\mathbf{x}} \in \mathbb{R}^p$ from the latent vector:

$$\hat{\mathbf{x}} = g_\phi(\mathbf{z}).$$

The network is trained to minimize reconstruction error, often mean squared error:

$$\mathcal{L}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2.$$

Once trained, the encoder f_θ is used as a feature extractor: the latent code $\mathbf{z}$ becomes the new feature vector. [9], [11]

5.2 Nonlinear Feature Extraction

Because the encoder and decoder can use nonlinear activation functions and multiple layers, autoencoders can capture complex, nonlinear manifolds in the data. This makes them powerful for:

- Dimensionality reduction beyond linear methods like PCA.
- Extracting features from images, audio, and time series.
- Pretraining representations for downstream tasks when labels are scarce.

Variants such as convolutional autoencoders, denoising autoencoders, and variational autoencoders (VAEs) further extend these capabilities. [10], [11]

5.3 Advantages and Limitations

Advantages:

- Capture nonlinear structures and complex patterns.
- Learn task-agnostic representations that can be reused.
- Work well with large amounts of unlabeled data.

Limitations:

- Require more data and compute than simple linear methods.
- Latent features are usually hard to interpret.
- Training can be unstable and sensitive to hyperparameters.

6 Other Feature Extraction Techniques (Overview)

Beyond PCA, LDA, and autoencoders, many other feature extraction methods exist. [12]

- **Kernel PCA:** applies PCA in a nonlinear feature space defined by a kernel function.
- **Independent Component Analysis (ICA):** seeks statistically independent components instead of uncorrelated ones.
- **t-SNE and UMAP:** primarily used for visualization by preserving local structure in 2D or 3D.
- **Domain-specific extractors:** SIFT or HOG for images, MFCCs for audio, word embeddings for text.

The best method depends on data type, dimensionality, and whether the task is supervised or unsupervised.

7 Practical Workflow and Tips

Integrating feature extraction into a machine learning pipeline typically involves:

1. **Preprocessing:** Clean data, handle missing values, and scale features when necessary.
2. **Choose a method:** For tabular data, start with PCA; for images or complex signals, consider autoencoders or convolutional networks.
3. **Fit extractor:** Learn the transformation ϕ on training data only.
4. **Transform data:** Apply ϕ to training and test sets to get extracted features.
5. **Train model:** Use the new features as inputs to a classifier or regressor.
6. **Evaluate:** Compare performance with and without feature extraction to verify its benefits.

Some good practices:

- Always fit the extraction model on training data to avoid data leakage.
- Standardize or normalize features before applying methods like PCA.
- For autoencoders, monitor both reconstruction error and downstream task performance.

8 Conclusion

Feature extraction is a powerful way to transform raw inputs into compact, informative representations that are easier for machine learning models to work with. In this article, we have:

- Clarified the difference between feature selection and feature extraction.
- Presented PCA as a fundamental linear method for unsupervised dimensionality reduction.
- Introduced LDA as a supervised extraction technique that maximizes class separability.
- Discussed autoencoders as a flexible framework for nonlinear representation learning.

There is no universal best method. Simple linear techniques like PCA are often a good starting point, while deep autoencoder-based methods shine when data is complex and abundant. In practice, it is common to experiment with several approaches and choose the one that offers the best trade-off between performance, interpretability, and computational cost. [6], [12]

References

[1] GeeksforGeeks, *Feature selection vs. feature extraction*, https://www.geeksforgeeks.org/machine-learning/feature-selection-vs-feature-extraction/, 2023.

[2] V. Prabhu, *Feature selection vs feature extraction: Machine learning*, https://vitalflux.com/machine-learning-feature-selection-feature-extraction/, 2024.

[3] Statsig, *Feature selection vs. feature extraction: Key differences*, https://www.statsig.com/perspectives/feature-selection-vs-extraction, 2025.

[4] Applied AI Course, *Feature extraction in machine learning*, https://www.appliedaicourse.com/blog/feature-extraction-in-machine-learning/, 2024.

[5] GeeksforGeeks, *Mathematical approach to pca*, https://www.geeksforgeeks.org/machine-learning/mathematical-approach-to-pca/, 2025.

[6] T. de Ridder, *Feature extraction using pca*, https://www.visiondummy.com/2014/05/feature-extraction-using-pca/, 2014.

[7] H. Niu et al., "Exploring unsupervised feature extraction algorithms," *Scientific Reports*, 2025.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[10] J. Qian et al., "A review on autoencoder based representation learning for industrial applications," *Engineering Applications of Artificial Intelligence*, 2022.

[11] Hex, *Using autoencoders for feature selection*, https://hex.tech/blog/autoencoders-for-feature-selection/, 2023.

[12] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection and feature extraction methods used on microarray data," *Advances in Bioinformatics*, 2007.